Diffusing ideas

Adventures with noise and building mathematical toys

Robert J. Hardwick

December 21, 2022

Introduction

Diffusing ideas has been written to explore a collection of interrelated ideas around the statistical inference and control of an extremely general class of stochastic phenomena. In order to manifest these ideas into reality, this project has also led to designing and building some new, open-source, scientific software tools that form a foundation from which one can build a variety of new applications. Testing and experimentation has also given us the opportunity to develop and play with a range of fun and interesting mathematical toy models which are motivated by the real world — the potential examples are vast, and we encourage the reader to think of more!

Any software that was developed in this book will always have been shared under a MIT license in the following public git repository for anyone to use: https://github.com/umbralcalc. All of this sofware should also still be considered as very much 'in development' and so it is advised that it is not deployed in production without careful scrutiny at the user's own risk.

No quest would be complete without a map, so in this introductory section we will now outline some of the key milestones of the project and their motivations within the context of the overall research ideas.

1. How do we generate a very general set of stochastic phenomena?

2. How do we learn/identify these phenomena from real-world data?

3. How do we generate a very general set of control policies to interact with these phenomena?

4. How do we then learn the optimal policies to achieve a specified control objective?

Table of contents

Building the 'stochadex'

Concept. The idea here is

1.1   Core mathematical formalism

Ideally, the stochadex sampler should be designed to try and maintain a balance between performance and flexibility of utilisation (it's meant to be a general stochastic modelling tool after all!). But before we dive into the software, we need to define the mathematical approach we're going to take. It seems reasonable to start with writing down the following formula which describes iterating some arbitrary process forward in time (by one finite step) and adding a new row each to some matrices $S' \to S$ and $X' \to X$

$$X_{t+\delta t}^i = F_{t+\delta t}^i(X', t) + S_{t+\delta t}^i(X', S', t), \tag{1.1}$$

where to explain things in a bit more detail, let's define

δt as the size of the next timestep

i as an index for the dimensions of the 'state' space

t as the current time which may either correspond to the index for some arbitrary stepping process (where $\delta t = 1$), or the explicit time variable for a discrete-time process or some discrete approximation to a continuous-time process

X as the next version of X' after one timestep (and hence one new row has been added)

S as the next version of S' (similarly to X and X')

$F^i_{t+\delta t}(X', t)$ as the latest element of some matrix which very generally can depend on X', t and δt

$S^i_{t+\delta t}(X', S', t)$ as the recorded values of a stochastic process defined over the dimensions of the state space indexed by i which may either vary with continuous sample paths in time or not, and which very generally can depend on X', S', t and δt.

So the idea here is to iterate the matrices X and S forward in time by a row, and use their previous versions (X' and S') as entire matrix inputs into functions which populate the elements of their latest rows. Note that I've avoided using the time variable t and its step size δt (reducing the formalism to that of an arbitrary stepping process) to try and make the diagram above more clear, but, more generally, these values may be present in the calculations of new matrix rows.

Why go to all this trouble of storing matrix inputs for previous values of the same process? For a large class of stochastic processes this memory of past values is essential to consistently construct the sample paths moving forward. This is true in particular for *non-Markovian* phenomena (see, e.g., Markov chains to get a sense of their antithesis), where the latest values don't just depend on the immediately previous ones but can depend on values which occured much earlier in the process.

1.2   Flavours of noise with continuous sample paths

For *Wiener process noise*, adopting the Itô interpretation in this section, we can define W^i_t is a sample from a Wiener process for each of the state dimensions indexed by i and our formalism becomes

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + W^i_{t+\delta t} - W^i_t \,. \tag{1.2}$$

Other interpretations of the noise are less immediately compatible with our formalism as it is currently written, e.g., Stratonovich or others within the α-family, but it seems less necessary to complicate the details of this section further, so we'll just cover these extensions at the software implementation level. Note also that we may also allow for correlations between the noises in different dimensions.

For *Geometric Brownian motion noise*, we simply have

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + X^i_t(W^i_{t+\delta t} - W^i_t) \,. \tag{1.3}$$

And say, e.g., *fractional Brownian motion noise*, where $B^i_t(H_i)$ is a sample from a fractional Brownian motion process with Hurst exponent H_i for each of the state dimensions indexed by i, we simply substitute again

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + B^i_{t+\delta t}(H_i) - B^i_t(H_i) \,. \tag{1.4}$$

Generalised continuous noises would take the form

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + g^i_{t+\delta t}(X', W^i_{t+\delta t} - W^i_t, \dots) \,, \tag{1.5}$$

where $g^i_{t+\delta t}(X', W^i_{t+\delta t} - W^i_t, \dots)$ is some continuous function of its arguments which can be expanded out with Itôs Lemma.

1.3 Flavours of noise with discontinuous sample paths

Jump process noises generally could take the form

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + J^i_{t+\delta t}(X', \dots) \,, \tag{1.6}$$

where $J^i_{t+\delta t}(X', \dots)$ are samples from some arbitrary jump process (e.g., compound Poisson) which could generally depend on a variety of inputs, including X'.

Poisson process noises would generally take the form

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + N^i_{t+\delta t}(\lambda_i) - N^i_t(\lambda_i) \,, \tag{1.7}$$

where $N^i_t(\lambda_i)$ is a sample from a Poisson process with rate λ_i for each of the state dimensions indexed by i. Note that we may also allow for correlations between the noises in different dimensions.

Time-inhomogeneous Poisson process noises would generally take the form

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + N^i_{t+\delta t}(\lambda^i_{t+\delta t}) - N^i_t(\lambda^i_t) \,, \tag{1.8}$$

where λ^i_t is a deterministically-varying rate for each of the state dimensions indexed by i.

Cox (doubly-stochastic) process noises would generally take the form

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + N^i_{t+\delta t}(\Lambda^i_{t+\delta t}) - N^i_t(\Lambda^i_t) \,, \tag{1.9}$$

where the rate Λ^i_t is now a sample from some continuous-time stochastic process (in the positive-only domain) for each of the state dimensions indexed by i.

Self-exciting process noises would generally take the form

$$X^i_{t+\delta t} = F^i_{t+\delta t}(X', t) + N^i_{t+\delta t}[\mathcal{I}^i_{t+\delta t}(N', \dots)] - N^i_t[\mathcal{I}^i_t(N'', \dots)] \,, \tag{1.10}$$

where the stochastic rate $\mathcal{I}^i_t(N', \dots)$ now depends on the history of N' explicitly (amongst other potential inputs - see, e.g., Hawkes processes) for each of the state dimensions indexed by i.

Generalised probabilistic discrete state transitions would take the form

$$X^i_{t+\delta t} = /F^i_{t+\delta t}(X', t) + T^i_{t+\delta t}(X') \,, \tag{1.11}$$

where $T^i_{t+\delta t}(X')$ is a generator of the next state to occupy. This generator uses the current state transition probabilities (which are generally conditional on X') at each new step.

1.4 Summary of desirable features

- using the learnings from the previous sections looking at specific example processes

- above formalism is so general that it can do anything - so while it shall serve as a useful guide and reference point, it would be good here to go through more of the specific desirable components we want to have access to in the software itself

- it might not always be convenient to have the windowed histories stored as S but some other varying quantity which can be used to construct S? take fractional brownian motion as an example of this! hence, need to provide more possible input histories into S

- want the timestep to have either exponentially-sampled lengths or fixed lengths in time

- formalism already isn't explicit about the choice of deterministic integrator in time

- but also want to be able to choose the stochastic integrator in continuous processes (Itô or Stratonovich?)

- enable correlated noise terms at the sample generator level

- configurable setup of simulations with just yamls + a single .go file defining the terms

Test cite [1]

1.5   Software design choices

Building a Rugby simulator game

Concept. The idea here is

2.1 Introduction

Since the basic game engine will run using the stochadex sampler, the novelties in this project are all in the design of the rugby match model itself. And, in this instance, I'm not especially keen on spending a lot of time doing detailed data analysis to come up with the most realistic values for the parameters that are dreamed up here. Even though this would also be interesting.

One could do this data analysis, for instance, by scraping player-level performance data from one of the excellent websites that collect live commentary data such as rugbypass.com or espn.co.uk/rugby.

This game is primarily a way of testing out the interface of the stochadex for other users to build projects with. This should help to both iron out some of the kinks in the design, as well as prioritise adding some more convenience methods for event-based modelling into its code base.

2.2 Designing the event simulation engine

We need to begin by specifying an appropriate event space to live in when simulating a rugby match. It is important at this level that events are defined in quite broadly applicable terms, as it will define the state space available to our stochastic sampler and hence the simulated game will never be allowed to exist outside of it. So, in order to capture the fully detailed range of events that are possible in a real-world match, we will need to be a little imaginative in how we define certain gameplay elements when we move through the space.

The diagrams below sum up what should hopefully work as a decent initial approximation while providing a little context with specific examples of play action.
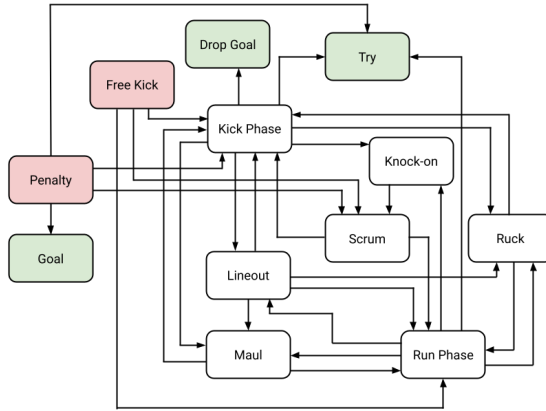

Figure 2.1: Simplified event graph of a rugby union match.

2.3   Linking to player attributes

2.4   Deciding on gameplay actions

2.5   Writing the game itself

Figure 2.2: Optional model ideas.

Bibliography

Bibliography

[1] E. Dimastrogiovanni, M. Fasiello, R. J. Hardwick, H. Assadullahi, K. Koyama and D. Wands, *Non-Gaussianity from Axion-Gauge Fields Interactions during Inflation*, 1806.05474.